Optimiser et Répartir ses Applications Mobiles avec Macchiato

Nicolas Petitprez, Romain Rouvoy, Laurence Duchien

Optimiser et Répartir ses Applications Mobiles avec Macchiato

Nicolas Petitprez, Romain Rouvoy et Laurence Duchien

Inria Lille – Nord Europe,
LIFL - CNRS UMR 8022,
Université de Lille 1, France
{nicolas.petitprez,romain.rouvoy,laurence.duchien}@inria.fr

1 Introduction

De nos jours, les utilisateurs de terminaux mobiles souhaitent des applications de plus en plus personnalisées et riches en fonctionnalités et consommant de plus en plus de données. Ces nouveaux usages se traduisent par une surconsommation des ressources du périphérique (par exemple, utilisation du processeur et de la connexion réseau). Sur un périphérique mobile, ces ressources sont limitées, principalement par la capacité de la batterie et les limitations technologiques et contractuelles du réseau. Il faut donc minimiser l'utilisation des ressources du périphérique, tout en maximisant les performances de l'application.

Dans un contexte de mobilité, l'ensemble des traitements peut être effectué sur le téléphone mobile, mais cela sollicite fortement les ressources du périphérique. Il est possible de déléguer les traitements à un serveur distant, mais, dans ce cas, les performances sont liées à celles du serveur et à la qualité de la connexion réseau. En cas de coupure de connexion, l'application cesse totalement de fonctionner. Une solution est donc d'envisager de répartir dynamiquement les traitements entre le périphérique et des serveurs distants tout en garantissant les performances et en étant tolérant à l'absence de connectivité réseau.

Certains travaux présentent les techniques de répartition d'une application mobile entre un périphérique mobile et un serveur [2,3]. Il est ainsi possible d'améliorer les temps de réponse d'une application et de réduire sa consommation de ressources [4]. Malheureusement, les solutions de répartition des traitements mobiles existantes sont coûteuses à mettre en place, complexes à utiliser, et n'offrent pas de support pour permettre l'adaptation des applications à l'environnement d'exécution. De plus, dans les applications mobiles, cet environnement d'exécution évolue en permanence : niveau de charge de la batterie, présence de la géolocalisation, qualité de la connexion réseau. Il est important d'offrir un support aux développeurs pour pouvoir facilement adapter les traitements à cet environnement et les répartir.

2 La Plate-forme Macchiato

Nous présentons dans cette démonstration la plate-forme Macchiato [1], développée pour construire facilement des applications mobiles performantes, adaptables et pouvant être réparties tout en étant économes en ressources. Macchiato propose aux développeurs un langage embarqué qui permet de définir des acteurs répartis exploitant les standards du Web. Les acteurs sont des entités autonomes qui communiquent par envoi de message. Ce modèle asynchrone permet de répartir efficacement les traitements entre différentes machines. Notre plate-forme présente des propriétés importantes qui permettent de répondre aux problématiques énoncées précédemment. Dans la suite, nous détaillons ces propriétés.

Cette plate-forme est **agnostique** vis à vis de l'environnement d'exécution. Les applications développées s'exécutent sans modification sur tous les environnements que la plate-forme supporte. L'utilisation d'un langage de script courant permet de cibler un nombre important d'architectures

1. Projet Macchiato : http://www.macchiato.fr/

d'exécution. Une abstraction de l'environnement permet d'avoir un code identique quel que soit l'environnement d'exécution. Il est actuellement possible d'exécuter des acteurs MACCHIATO dans les navigateurs Web, dans les applications mobiles *Android* et sur les serveurs d'application *vert.x* [2].

Pour pouvoir s'adapter facilement aux besoins et à l'environnement, la plate-forme est **réflexive**. Elle permet d'introspecter l'architecture de l'application et de reconfigurer son fonctionnement par des opérations d'ajout, de modification ou de suppression d'acteurs.

Pour répondre aux problématiques de performance et d'économie de ressources, la plate-forme est **auto-optimisable**. La figure 1 présente le système d'optimisation de la plate-forme MACCHIATO. Lors de l'exécution, la plate-forme collecte des données sur le fonctionnement de l'application, comme le nombre et la taille des messages échangés, l'utilisation de la connexion réseau, etc. Les informations collectées permettent de modéliser le problème d'optimisation comme un problème pseudo-booléen [1]. L'utilisation d'un solveur permet alors de trouver la meilleure répartition de l'application en fonction de l'objectif retenu. Il est, par exemple, possible de limiter la quantité de données qui transite par la connexion réseau du périphérique.

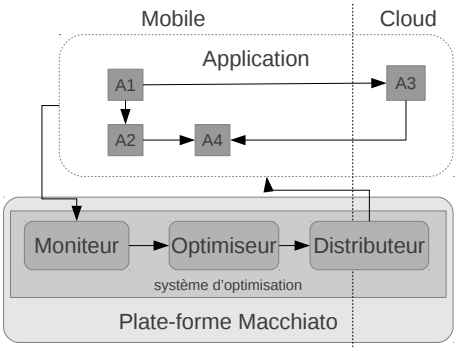

FIGURE 1. Système d'optimisation de la plate-forme MACCHIATO

Dans un contexte mobile, il est fréquent d'avoir des problèmes de qualité de connexion. La plate-forme se doit donc d'être **robuste**. Le modèle de l'application ainsi qu'une copie des acteurs déployés à distance sont conservés sur le périphérique du client. En cas de défaillance d'un serveur distant ou de la connectivité du périphérique, la plate-forme redirigera automatiquement les messages à destination du serveur inaccessible vers les acteurs fonctionnant sur le périphérique du client.

Dans notre démonstration, nous utiliserons le langage embarqué de la plate-forme MACCHIATO pour construire une application mobile. Nous montrerons ensuite comment la plate-forme surveille l'exécution de l'application, planifie son optimisation, et la répartit entre l'environnement du client et un serveur d'application.

Références

1. Daniel Le Berre and Anne Parrain. The Sat4j library, release 2.2 system description. *Journal on Satisfiability, Boolean Modeling and Computation*, 7(2010) :59–64, 2010.
2. Ioana Giurgiu, Oriana Riva, and Gustavo Alonso. Dynamic software deployment from clouds to mobile devices. In *Middleware*, pages 394–414, 2012.
3. Roelof Kemp, Nicholas Palmer, Thilo Kielmann, and Henri Bal. Cuckoo : a computation offloading framework for smartphones. *Mobile Computing, Applications, and Services*, pages 59–79, 2012.
4. K Kumar and YH Lu. Cloud computing for mobile users : Can offloading computation save energy ? *IEEE Computer*, (April) :51–56, 2010.

2. vert.x : `http://vertx.io`